ALGORITHMS FOR THE VEHICLE ROUTING PROBLEM

By

ANIL KUMAR LAWANIA

INDUSTRIAL & MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MAY, 1992

ALGORITHMS FOR

THE VEHICLE ROUTING PROBLEM

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

by

ANIL KUMAR LAWANIA

to the

Industrial and Management Engineering Programme

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 1992

C E R T I F I C A T E

This is to certify that the work contained in the thesis entitled **"ALGORITHMS FOR THE VEHICLE ROUTING PROBLEM"**, by Anil Kumar Lawania, has been carried out under my supervision and that this has not been submitted elsewhere for the award of a degree.

[illegible]

(Dr. R.K. Ahuja)
Asst. Professor
Industrial and Management Engineering Programme
Indian Institute of Technology
Kanpur - 208 016 (U.P.)

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and indebtedness to my guide Dr. Ahuja for his valuable suggestions, advices, and constant encouragement during my thesis work. I once again thank him for the extra efforts made by him for the completion of this thesis in time.

I convey my sincere thanks to all faculty members and my colleagues at CMC Ltd who have cooperated and helped me at several times.

I am grateful to my wife JYOTSNA for her support, encouragement and cooperation which I always received while doing M. Tech.

A. K. Lawania

ABSTRACT

The Vehicle Routing Problem (VRP) is the problem of designing routes (sequence of visiting customers), for a set of vehicles stationed at depots, to visit a set of given customers. These routes must satisfy given constraints. The routing of vehicles is an area of importance to both operation researchers and transportation planners. The VRP, even in it's simplest form, is computationally complex and is a NP-complete problem for which no polynomial bounded algorithm is likely to exist. However, several approximation algorithms/heuristics have been developed, which perform very well. Most of these heuristics consider single depot problem. There are many practical problems in which vehicles are stationed at several depots and hence multi-depot VRP arises. In this thesis, an attempt have been made to develop heuristic algorithms to solve the multi-depot vehicle routing problem.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The Vehicle Routing Problem (VRP) is the problem of designing routes (sequence of visiting customers), for a set of vehicles stationed at depots, to visit a set of given customers. These routes must satisfy given constraints like total demand of the customers in a route should not exceed the capacity of the vehicle. The routing of vehicles is an area of importance to both operation researchers and transportation planners. There are many variations of the problem ranging from school bus, collection of mail boxes, telephone coinboxes, routing of delivery trucks, despatching of consumer goods and assignment, and routing of service engineers to visit customers. In all cases, the basic components of the problem are a fleet of vehicles with fixed capabilities and a set of demand points for transporting certain objects.

The VRP, even in it's simplest form, is computationally complex and is a NP-complete problem for which no polynomial bounded algorithm is likely to exist. However, several approximation algorithms/heuristics have been developed, which perform very well. Most of these heuristics consider single depot problem. While considering assignment of service engineers tc customers for servicing machines, multi-depot VRP arises becaus service engineers are located not only at central office but als at several service centres. In this thesis, an attempt have bee made to develop heuristic algorithms to solve the multi-depo vehicle routing problem which perform very well on many test problems.

1.2 PROBLEM DESCRIPTION

The vehicle routing problem in the most general sense can be defined as follows : A set of customers, each with a known location and a known requirement for some commodity, is to be supplied from a set of depots by a set of delivery vehicles of known capacity. The objective of the solution may be stated as cost minimization (distribution costs and vehicle or depot costs), or service improvement (increasing distribution capacities, reducing distribution time).

A vehicle route is a sequence of pick up and/or delivery points which the vehicle must traverse in order, starting and ending at a depot or domicile. The routing decision involves determining which of the demands will be satisfied by each vehicle and what route each vehicle will follow in servicing its assigned demand. Vehicle routes must also satisfy a variety of following constraints as per the problem requirement:

(a) The requirements of all the customers must be met and each customer should be visited by only one vehicle.

(b) The node constraints of vehicles must not be violated (i.e., the number of nodes/customer allocated to each vehicle must not exceed some predetermined number).

(c) The cost (alternatively, time or distance) constraints of vehicles must not be violated (i.e., the total cost for each vehicle to complete its tour must not exceed some predetermined level).

(d) The load constraints of the vehicle must not be violated (i.e., the total load allocated to each vehicle must not exceed its capacity).

Most practical point to point vehicle routing problems have the following characteristics:

1. A set of customers x_i: i =1 to N, having demand q_i, where N is the number of customers.

2. A set of depots where vehicles are stationed.

   Single depot : Depot x_o

   Multiple depots : Depot x_i : N + 1 to N + M,

   where M is the number of depots.

   In case of multi-depot, each depot may have an upper bound on the number of vehicles that can be housed there.

3. Symmetrical distances C_{ij} between all pairs of customers and depots.

$$C_{ij} \; : \; \begin{array}{l} i = 1 \text{ to } N + M \\ j = 1 \text{ to } N + M \end{array}$$

4. Vehicle capacity for all vehicles.

   All vehicle may have same capacity Q or vehicles may have different capacities Q_l, l = 1 to K, where

   K is the number of vehicles.

5. Maximum length of a route allowed is L and/or maximum travel time allowed for a route is T.

6. Some locations may have time windows [ST_i, FT_i] during

which only it should be visited.

ST_i : Start time for x_i,
FT_i : Finish time for x_i.

1.3 APPLICATIONS

1.3.1 School Bus Routing

In the school bus routing problem, there are a number of schools and each one has a set of associated bus stops. In addition, there are a given number of students associated with each bus stop. Each school has a fixed start and finish time with a time window about each of these for the delivery of students to the school in the morning and the pick up of students from the school in the afternoon. The problem is to minimize the number of the buses used or total transportation costs while servicing all the students and satisfying all the time windows.

1.3.2 Subscriber Dial-a-Ride Routing

In the subscriber dial-a-ride environment, customers call in advance requesting service. Each customer specifies a pick up point, a delivery point and a desired time of pick up and/or delivery. The problem is to develop a set of routes for the vehicles in a fleet of fixed size in order to minimize total customer dissatisfaction or inconvenience.

1.3.3 Mail Pickup problem

We are given the network of post offices. Each stop represents a post office to be serviced within a fixed time span. For this application, the violation of the lower bound of the time window is tolerated and implies only an additional waiting time for the vehicle. For each vehicle, the speed and maximal travel time is specified. Number of the vehicles and total travel distance/time is to be minimized.

1.3.4 Garbage Truck Routing

Garbage is to be collected from various points and to be dumped at specified location. Collecting trucks have fixed capacity. Objective is to design routes such that the transportation cost is minimized.

1.3.5 Milk Distribution

Milk is to be distributed from the depot or set of depots to the booths located in various parts of the city. Each distribution truck has fixed capacity and each booth has certain demand for milk. Objective is to minimize the total transportation cost for satisfying demands of all the booths while considering restriction on the vehicle capacity and maximum allowed tour length.

1.3.6 Assignment of service (maintenance) engineers

A computer maintenance organization (or any maintenance organization) has large number of contracts for repairing of computers installed at customers site. These customers are geographically distributed in the whole city. There are few service centres which are located in various parts of the city. Whenever there is a break down of computer, the customer informs at central office of the maintenance company. Central office assigns these calls to engineers preferably from the nearest service centres. Normally large number of calls, requesting service, are received in a day. An engineer is assigned several customers to visit in a day. Visiting all the customers involves large amount of travelling, which is a major component of company expenses. Objective is to give each engineer such customers so that total travelling cost by all the engineers is minimised.

1.4 MOTIVATION OF THESIS

The author is working with CMC Ltd whichis a premier computer maintenance company of the country. It has a large number of service contracts for repairing computers at customer's site. Service engineers stay at a central office and service centres which are located in areas where number of customers are significant. Call-Despatch receives all the complaints from customers, who require service. Call-Despatch assigns these calls to service engineers and normally more then one call is assigned to each engineer. In metropolitan cities like Delhi and Bombay, engineers have to travel very large distances every day which is done by hired taxies. Cost of travel is going higher and higher and is a significant portion of the expenses of the company. This traveling cost is to be minimized in order to save travel expenses.

The structure of the problem is clearly similar to that of a vehicle routing problem. Each engineer can be considered as a vehicle of capacity 8 hours and customers having demands which are representative of their required repair times. The route for each engineer should be decided such that all customers are attended by only one engineer and the total travel cost is minimized. Since engineers stay at central office and service centres, it is a case of a multi-depot VRP. Since most of the VRP heuristics deals with the single depot problem, an heuristic for the multi-depot VRP have been developed and presented in this dissertation.

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

This chapter describes the existing algorithms for solving VRP's starting from Travelling Salesman Problem (TSP). TSP is the most fundamental problem and all vehicle routing problems are extensions of TSP and multiple-TSP. Since TSP itself is a NP-complete problem, there is no polynomial-time exact algorithm for the VRP. Most of the work has been done for the development of approximate algorithms, called heuristic algorithm. There have been different approaches for the development of heuristics and classifications of VRP heuristics have been presented based on route building/development approaches.

Some of the most popular heuristic algorithms for the VRP are : The Clarke and Wright's [1964] approach which is one of the most widely accepted approaches. In this approach those two customers are linked whose linking results in maximum saving at each iteration. Mole and Jameson [1976] uses different function for the calculation of savings. Gillet and Miller [1974, 1976] has given a method known as Sweep algorithm. In this method, starting routes are formed by arranging customers according to increasing values of their polar coordinates. Christofides [1979] also uses savings criteria at each iteration. Fisher and Jaikumar [1981] has presented a generalized assignment based heuristic. Ahn Byong Hun and Shin Jai [1991] has described in detail how to incorporate time window constraints. In this chapter we present a brief description of these algorithms and their various modifications and extensions.

2.2 TRAVELLING SALESMAN PROBLEM

The travelling salesman problem is perhaps the most famous problem in the area of network and combinatorial optimization; its simplicity and yet its difficulty has made it an alluring problem that has attracted the attention of many noted researchers over a period of several decades. The problem is deceptively easy to state: starting from ahome base, say node 1, a salesman wishes to visit each of several cities, represented by nodes 2,..,.n, exactly once and return to the home base, doing so at the lowest possible cost. Given the graph G = (N,A) and internode cost matrix C, the TSP concerns finding the minimum cost cycle in G that visits every node once. The travelling salesman problem is a generic core model that captures the combinatorial essence of most routing problems and indeed most other routing problems are extensions of it. The TSP is an NP-complete problem and, hence, a polynomial time exact algorithm is not likely to exist (Karp [1972]). But there are several approximation algorithms / heuristics whose computation time growth rate is a low order polynomial in n and which have been experimentally observed to perform well.

The TSP is closely related to the VRP. In the vehicle routing problem, a set of vehicles, each with a fixed capacity, must visit a set of customers to deliver (or pick up) a set of goods. The best possible set of delivery routes are to be determined. Once a set of customers are assigned to a vehicle, that vehicle should take the minimum cost tour through the set of customers assigned to it; that is, it should visit these customers along an optimal travelling salesman tour.

2.3 MULTI-TRAVELLING SALESMAN PROBLEM

The Multi-Travelling Salesman Problem (Oriff [1974], Garcia Diaz [1985]) is an extension of TSP, where number of salesmen is more than one. Given N cities and M available salesmen based at one of these cities, it is desired to visit the N cities in such a way that (a) each city different from the base city is visited exactly once; (b) all tours start and finish at the base city; and (c) each tour is assigned to exactly one salesman. An optimal solution to this problem consists of one or more tours resulting in minimal total travel cost. Most of the real world vehicle routing problem have upper limit on the capacity of the vehicles, therefore number of vehicles required tc meet all the customer demand, will be more than one. Deliver} problem, which is an M-vehicle routing problem (M-VRP) is basicall} an M-TSP subject to Vehicle Capacity constraints and unknow (variable) M.

Any M-TSP can be solved as an equivalent TSP on transformed network (Bellmore and Homg [1974]). Thi transformation requires solving a TSP on a network with N+M nodes Since M << N generally, this is not a substantial increase i problem size. Hence the best M-TSP algorithm could be expected t be as good as the best TSP algorithm. A number of algorithms ha been developed to solve the M-TSP problem. Bellmore and Ho [1974] showed that an N city, M-Salesman problem could transformed to an equivalent TSP problem with N+M-1 citie Svetska and Huckfeldt [1973] used a similar transformation and relaxing the subtour elimination constraints, they solved proble involving upto 60 cities and 10 salesmen. Ali and Kenningt [1980] developed a Lagrangian relaxation of the degree constrai and used a minimal m-tree algorithm to solve small number of M-TSP's. Laporte and Nobert [1979] developed an algorithm for the MTSP based on a relaxation of the subtour elimination and integrality constraints and then used linear programming techniques to solve some small to medium size M-TSP's. Garcia-Diaz [1985] gives a heuristic circulation network approach to solve the M-TSP.

2.4 VEHICLE ROUTING PROBLEM

The vehicle routing problem, in its most simplest form, is stated as follows : Given (i) a fleet of vehicles of capacity Q domiciled at a common depot, say node 1; (ii) a set of customers sites $j = 2,..,n$, each with a prescribed demand q_j; and (iii) a cost C_{ij} of travelling from location i to location j. What is the minimum cost set of routes for delivering (picking up) the goods to the customer sites?

The VRP even in the simplest form is computationally complex. Since the TSP is a special case of the VRP, VRP is alsc a NP-complete problem. This observation suggests that th algorithmic possibilities for optimization methods are limited an must be restricted to designing effective heuristic algorithms Lenstra and Rinnoy Kan [1981] presents a summary of computationa complexity as applied specifically to vehicle routing ar scheduling problems. They show that most of these problems ai NP-complete. Consequently, VRP appears to be a prime candida for analysis by heuristic methods. Since 1959, when Dantzig a Ramser [1959] first introduced the vehicle routing problem a proposed a linear programming based heuristic for its solutio Subsequently, the overwhelming majority of attempts at solving t problem have focused on heuristic methods. As Christotides [1976], who has been a leading proponent of exacts algorithms, has acknowledged, the largest vehicle routing problem of any complexity solved by exacts methods does not contain more then 30-40 customers. Magnanti [1981] presents an excellent survey of vehicle routing algorithms.

2.5 CLASSIFICATION OF VRP ALGORITHMS

Past work on VRP has been concerned almost exclusively with heuristics. Most solution strategies/ heuristics for vehicle routing problems can be classified as one of the following approaches.

(a) **Construction -** Construction algorithms generate a solution by adding a link between two customers sequentially until all customers have been assigned to some route. Each time a link is added, constraints are checked for violation. The choice of a link is motivated by some measure of cost savings. The most often used tour building heuristic is Clarke and Wright method [1964]. Gaskell [1967] and Yellow [1970] introduced the concept of modified savings. Golden et al. [1977], Norback and Love [1977], Hinson and Mulherkar [1975] also use construction approaches.

(b) **Cluster first-route second -** These procedures group o cluster nodes first and then design economical routes ove each cluster as a second step. Gillett and Miller [1974] Gillett and Johnson [1976] and Karp [1977] use thes approaches.

(c) **Route first-cluster second** - These procedures work in reverse sequence. First a large (usually infeasible) route or cycle ıs constructed which ıncludes all of the demand entities. Next, the large route ıs partitioned into number of smaller, but feasıble routes. Newton and Thomas [1974], Bodin and Berman [1979] and Bodın and Kursh [1978] use these approaches.

(d) **Improvement** - Improvement algorithms start wıth a feasible solution and then successively improve ıt by a sequence of exchanges or mergers. Well-known branch exchange heurıstıcs are developed by Lin and Kernighan [1973], Chrıstofıdes and Eilon [1969] and Russel [1977].

(e) **Mathematical Programming** - Mathematical programming approaches start wıth a mathematical programming formulation and an exact solution procedure and then modify the solution procedure to obtain an effıcient heuristic algorıthm. Fisher and Jaıkumar [1981] use this approach. Christofıdes et al [1981] , Stewart and Golden [1979] dıscuss Lagrangean relaxation procedures for the routing of vehicles.

(f) **Exact procedures** - Exact procedures for solving vehicle routing problems include specialized branch and bound anc cutting plane algorithms. Some of these are described b Held and Karp [1971], Crowder and Padberg [1980], anc Christofides et al.[1981].

2.6 DESCRIPTIONS OF ALGORITHMS

Following are the details of some well-known heuristi algorithms for the VRP.

2.6.1 Saving Algorithm of Clarke and Wright [1964]

This algorithm is the most widely known heuristic algorithm and has formed the basis of many heuristics. The algorithm proceeds as follows

Step 1 - Calculate the savings $s_0(i,j)$ for all pairs of customers x_i, x_j. The saving of a customer x_i with respect to x_0 and x_j is given by

$s_0(i,j) = c_{i0} - c_{ij} + c_{0j}$

Step 2 - Arrange the savings in the descending order.

Step 3 - Starting from the top of the list do either of the following versions:

Parallel version

(i) If linking of customers corresponding to maximum saving, results in a feasible route according to the constraints of the VRP, then add this link to the solution; if not, reject the link.

(ii) Try the next link in the list and repeat (i) until no more links can be chosen.

Sequential Version

(i) Find the first feasible link in the list which can be used to extend one of the two ends of the currently constructed route.

(ii) If the route can not be expanded, or no route exists, choose the first feasible link in the list to start a new route.

(iii) Repeat (i) and (ii) until no more links can be chosen.

Step 4 - The links picked form a solution to the VRP.

2.6.2 Algorithm of Mole and Jameson [1967]

The algorithm of Mole and Jameson is a sequential procedure in which, for a given value of the two parameters λ and μ, the following two criteria are used to expand a route under construction :

$$e(i,l,j) = C_{il} + C_{lj} - \mu C_{ij}$$

$$\sigma(i,l,j) = \lambda C_{0l} - e(i,l,j)$$

The algorithm then proceeds as follows :

Step 1 - For each unrouted customer x_l compute the feasible insertion in the emerging route R as

$e(i_l,l,j_l) = \min [e(r,l,s)]$

for all adjacent customers $x_r, x_s \in R$ where x_{i_l} and x_{j_l} are customers between which x_l has the best insertion.

Step 2 - The best customer x_{l*} to be inserted in the route is computed as the one for which the following expression is maximized :

$\sigma(i_{l*}, l*, j_{l*}) = \max [\sigma(i_l,l,j_l)]$

for x_l unrouted and feasible.

Step 3 - Insert x_{l*} in route R between x_{il*} and x_{jl*}.

Step 4 - Optimize route R using r-optimal methods.

Step 5 - Return to Step 1 to start a new route R, either until all customers are routed or no more customers can be routed.

2.6.3 Sweep Algorithm of Gillet and Miller [1974, 1976]

This algorithm requires geographical coordinates for each customer.

Step 0 - Customers are ordered according to increasing values of their polar coordinate angles $\theta_i = \theta_o[i,1]$, taking customer 1 as the reference. The construction of an emerging route R is described below.

Step 1 - Starting from the unrouted customer x_{i1} with the smallest polar coordinate angle, include in the route consecutive customers x_{i2}, x_{i3},.. as long as the set[x_{i1}, x_{i2},...] of customers included can be routed into a feasible route R. The emerging route R is optimized by applying 2-opt or 3-opt (Lin and Kernighan [1973]) procedures. Let x_l be the last customer that has entered R.

Step 2 - Find the unrouted x_j nearest to customer x_l of the route. If the insertion of x_j in the route is feasible and worthwhile, insert x_j in the route, update x_l and repeat step 2.

Step 3 - Find the best customer x_r to be removed from the route, if the replacement of x_r by x_j is feasible and worthwhile, remove x_r from the route, insert x_j in it, update x_l and go to step 2.

Step 4 - Find the unrouted customer x_s nearest to x_j. If the replacement of x_r by x_j and x_s is feasible and worthwhile, remove x_r from the route, insert x_j and x_s

in it, update x_l and go to step 2. Otherwise the route is complete.

Step 5 - Return to step 1 repeat until either all customers are routed or no more customers can be routed.

2.6.4 Two Phase algorithm of Christofides et al [1979]

This algorithm consist of two phases. In the first phase, a solution is produced in a sequential way according to the following steps :

Phase 1

Step 1 - Set $h = 1$

Step 2 - Choose an unrouted customer x_{i_h} to initialize an emerging route R_h. Compute for all the unrouted customers x_l the following score

$$\delta_l = C_{0l} + \lambda C_{li_h}, \qquad \lambda \geq 1$$

Step 3 - Insert into R_h the feasible customer x_{l*} such that

$\delta_{l*} = \min[\delta_l]$, $\quad x_l$ unrouted and feasible.

Optimize route R_h using r-optimal methods and repeat step 3 until no more customers can enter R_h.

Step 4 - Set $h = h+1$ and return to step 2 to start a new route R_h until all customers are routed or no more customers can be routed.

Phase 2

In the second phase, h emerging routes are considered where the rth route is $\overline{P}_r = (x_o, x_i, x_o)$ and x_i is the same customer chosen at step 2 in the first phase to initialize route R_r. Let S be the set of routes $\bar{R}_r = (x_o, x_l, x_o)$ r =1,..,h.

Step 1 - Compute for each unrouted customer x_l and for each route $\bar{R}_r \in S$.

$$\epsilon_{rl} = C_{0l} + \mu C_{li_r} - C_{0i_r} \quad \mu \geq 1.$$

Associate x_l with that route $\bar{R}_{r*}$ such that

$$\varepsilon_{r*l} = \min_{R_r \in S} [\varepsilon_{rl}].$$

Step 2 - Choose any route $\bar{R}_r \in S$, set $S = S - \bar{R}_r$ and compute for each customer x_l associated with this route the following score

$$\bar{\delta}_l = \epsilon_{r'l} - \epsilon_{rl}$$

where r′ is given by $\varepsilon_{r'l} = \min_{R_r \in S} [\varepsilon_{rl}]$

Step 3 - Insert into $\bar{R}_r$ the feasible customer x_{l*} associated with $\bar{R}_r$ such that $\delta_{l*} = \max [\delta_l]$, x_l feasible and associated with $\bar{R}_r$.

Optimize Route $\bar{R}_r$ using r-optimal methods and repeat step 3 until there are no more feasible customers associated with $\bar{R}_r$.

Step 4 - If $S \neq \phi$, go to step 1 of phase 2. Otherwise, if all the customers are routed, stop; or, if unrouted customers remain, go to step 1 of phase 1.

2.6.5 A generalized Assignment Heuristic of Fisher and Jaikumar [1981]

The basic idea of this approach can be described in terms of the following formulation of VRP as a nonlinear generalized assignment problem.

$$\min \sum_{k} f(y_k)$$

s.t.

$$a_i y_{ik} \le b_k \qquad k = 1\ldots,k$$

$$y_{ik} = \begin{cases} k & i = 0 \\ 1 & i = 1,..,n \end{cases}$$

$$y_{ik} = 0 \text{ or } 1 \quad i = 0,..,n$$
$$k = 1,..,k$$

where $f(y_k)$ is the cost of an optimal travelling salesman tour of the customer in $N(y_k)$ in $N(y_k)=\{i|y_{ik}=1\}$. Observe that $f(y_k)$ is an extremely complicated function and can not even be written down for nontrivial problems. This heuristic is based on constructing a linear approximation $\sum_{=1}^{n} d_{ik}y_{ik}$ of $f(y_k)$.

The heuristic begins with a set of seed customers $i_1,\ldots,i_k$ that are assigned to the vehicles $1,\ldots,k$, respectively. The coefficient d_{ik} is then set to the cost of inserting customer i into the route in which vehicle k travels from depot directly to customer i_k and back.

$$d_{ik} = \min\left[c_{0i} + c_{ii_k} + c_{i_k0},\ c_{0i_k} + c_{i_ki} + c_{i0}\right] - \left[c_{0i_k} + c_{i_k0}\right]$$

Seed customers can be selected either by an automatic rule or by a scheduler. In automatic selection, to determine $w_1,\ldots w_k$(the k seed points) the plane is partitioned into k cones corresponding to k vehicles. Then w_k is located on the ray bisecting cone k. To determine these cones, first the plane is partitioned into n smaller cones, one for each customer. The infinite half ray forming the boundary between two customer cones is positioned to bisect the angle formed by half rays through the two immediately adjacent customers. Associate the weight a_i with customer cone i and define

$$\alpha = \frac{\sum_{i=1}^{n} a_i}{K_b}$$

Each vehicle cone is then formed from a contiguous group of customer cones or fractions of customer cones. The weight of the group is required to be equal to αb. The point w_k is located along the ray bisecting the kth cone.

2.6.7 Savings Algorithm of Paessens [1988]

This algorithm uses the following saving function

$$S_{ij} = dp_i + dp_j - g*d_{ij}+f*|dp_i dp_j|$$

$$0 < g \leq 1$$

Successful parameter combinations (g,f) are]

$$g = 0.8\ (0.1)2.0 \text{ with } f = 0.0(0.1)1.0$$

2.6.8 VRP with Time Windows

This method deals with time windows and time varying congestion constraints in VRP.

Given N customers to serve with

e_i = the earliest service time at node i;

l_i = the latest service time at node i;

$\tau_{ij}(x)$ = the travel time from node i to node j via arc (i,j), given that trip starts from node i at time x;

$A_{ij}(y)$ = the arrival time at node j through arc (i,j) given that service begins at time y at node i, that is, $A_{ij}(y) = y + s_i + \tau_{ij}(y+s_i)$ where s_i is the constant service duration time for node i

Following variables are used for computation :

t_i = the time at which service begins for node i;

d_i = the effective latest service time at node i that allows us to maintain the feasibility of a current route.

The service start time at node j is constrained within a time window $[e_i, l_i]$. If a vehicle travels directly from node i to node j, $t_j = \max\left\{e_j, A_{ij}(t_i)\right\}$. The vehicles leave depot at time e_o and should return to the depots not later than l_o. The objective is to find the feasible routes with minimum total travel time.

It is assumed that for any two nodes i and j, and any two service start times x and y such that $x > y$, $A_{ij}(x) > A_{ij}(y)$, that is, earlier departure from node i guarantees earlier arrival at node j. Since $A_{ij}(.)$ is strictly increasing function, inverse function $A_{ij}^{-1}(x)$ is uniquely defined and is interpreted as the departure time at node i so that node j can be reached at time x.

Let (0,1,2,...m,0) denote a partial route, which is feasible if each node in the route is visited within its corresponding time window.

The effective latest service start time d_i at node i on a feasible route, is given from the following backward recursive relations :

$$d_i = \min\left\{l_i,\ A^{-1}_{i,i+1}(d_{i+1})\right\} \qquad 0 \le i \le m-1$$

$$d_m = \min\left\{l_m,\ A^{-1}_{m,0}\ (l_o)\right\}$$

Also actual service start times t_i's are trivially computed from the forward recursion $t_i = \max\left\{e_i,\ A_{i-1,i}(t_{i-1})\right\}$ starting from i=1.

Following feasibility conditions can be easily tested using d_i's and t_i's.

Inserting an unrouted node to a given route

When node u, an unrouted node, is inserted between two adjacent nodes, say, s and s+1, on a feasible partial route (0,1,2,..,m,0), the resulting new partial route remains feasible if $t_u \le d_u$ where

$$t_u = \max\left\{e_u,\ A_{su}(t_s)\right\}$$

$$d_u = \min\left\{l_u\ A^{-1}_{u,s+1}(d_{s+1})\right\}$$

This condition implies that the values of t_i's and d_i's of the original route, are not required to be updated, each time a new insertion location is tried.

Combining two distinct routes into one

When two distinct routes are merged into one by linking the last node i of one route into first node j of the other route, then newly combined route is feasible if $A_{ij}(t_i) \le d_j$.

Exchanging some node

R-optimal heuristics exchange the visitation sequence among nodes within a given route, in particular, between a string of adjacent nodes and some single node within a given route.

Let (0,1,2...m,0) is a route that is currently feasible. A part of this route, say, the path (y, y+1,...,z) such that $1 \le y \le z \le m$ is to be moved to other location on the route.

First, time windows along this path are tightened up as follows :

$$\begin{aligned} e'_y &= e_y \\ e'_i &= \max\left\{e_i,\ A_{i-1,i}\ (e'_{i-1})\right\} \quad y+1 \le i \le z \\ l'_z &= l_z \\ l'_i &= \min\left\{l_i,\ A^{-1}_{i,i+1}(l'_{i+1})\right\} \quad y \le i \le z-1 \end{aligned}$$

using $\left[e'_i,\ l'_i\right]$ in place of $\left[e_i,\ l_i\right]$, the time feasibility conditions are as follows :

(i) If a vehicle that departs node z at time e'_z cannot arrive at node u by l_u, that is, if $A_{zu}(e'_z) > l_u$, then the path (y,y+1...,z) cannot precede node u.

(ii) Symmetrically, if a vehicle that departs node u at time e_u cannot arrive at node by l'_y, that is, if $A_{uy}(e_u) > l'_y$, then node u cannot precede the path (y, y+1...z).

2.7 TEST PROBLEMS

Christofides et al [1979] has published data for some single depot vehicle routing problems. Many researchers have used these test problems to compare their algorithms with others. Paessens [1988] has published results of many well known algorithms using these test problems.

We could not find any test problem data for multi-depot VRP. Therefore, these test problems have been used to measure the performance of our algorithms for solving single depot VRP which is a special case of multi-depot VRP. Multi-depot test problems have been generated by adding more depots to these test problems.

2.7.1 Data for Test Problems

Problem 1 - This is a 50 customer single depot problem. Data is published by Christofides [1969] as Problem 8 on Page 317.

Problem 2 - This is a 75 customer single depot problem. Data is published by Christofides [1969] as Problem 9 on Page 317.

Problem 3 - This is a 100 customer single depot problem. Data is published by Christofides [1969] as Problem 10 on Page 318.

Problem 4 - This is a 150 customer single depot problem. It is generated by adding the customers of Problem 1 and 3 to the depot and vehicle list of Problem 3.

Problem 5 - This is a 199 customer single depot problem. It is generated by adding the customers of Problem 4 to the first 49 customers of Problem 2 with depot as for Problem 4.

Problem 6 - This is a 120 customer single depot problem. Data is published by Christofides [1979] as Problem 11 on Page 336.

Problem 7 - This a 100 customer single depot problem. Data is published by Christofides [1979] as Problem 12 on Page 337.

Problem 8 - This is a 199 customer 5 depot (multi-depot) problem. It is generated by adding to the Problem 5 following depot coordinates : $\left\{(20,20),\ (10,45),\ (40,50),\ (15,85)\right\}$. All the customers and depot of Problem 5 are retained as such.

Problem 9 - This is a 120 customer 5 depot (multi-depot) problem. It is generated by adding to the Problem 6 following depot coordinates : $\left\{(20,20),\ (40,50),\ (15,85),\ (65,10)\right\}$. All customers and depot of Problem 6 are retained as such.

Problem 10 -This is a 100 customer 2 depot (multi-depot) problem. It is generated by adding to the Problem 7 following depot coordinates : $\{(10,45)\}$.
All customers and depot of Problem 7 are retained as such.

Problem 11-20 - These are randomly generated problems as per follows.

Problem	Number of customers	Number of Depots
11	100	5
12	200	6
13	300	4
14	400	10
15	500	4
16	600	7
17	700	5
18	800	10
19	900	10
20	1000	20.

Vehicle capacities are 200 for problems 11-15 and 300 for problems 16-20.

2.7.2 Results of Test Problems

The comparative study of the solutions by different algorithms has done by Paessens [1988]. Results for Problem 1 to 7 have been published and are given in Table 2.1(a) and 2.1(b).

TABLE 2.1(a)

Method		P1	P2	P3	P4
Paessen Problem Ref.		C1	C2	C3	C8
Gillett and Miller	-	-	-	-	1079/12
Mole and Jameson	-	575/5	910/10	882/8	1259/12
Christofides et al (tree search)	-	534/5	871/11	851/8	1064/12
Christofides et al (2 Phase)	-	547/5	883/11	851/8	1093/12
	R	537/5	873/11	858/8	**1082/12**$^{+}$
Probol (Real)	-	**521/5***	858/10	831/8	1100/12
	R	**525/5**$^{+}$	859/10	843/8	1100/12
Fisher and Jai Kumar	-	524/5	857/10	833/8	1014/12*
Saving/P (Real)	-	580/6	905/11	869/8	1126/12
	R	585/6	907/10	889/8	1140/12
Nagel	R	528/5	**844/10**$^{+}$	**832/8**$^{+}$	
	-	523/5	**841/10***	827/8	
Rhode	-	526/5	883/11	**825/8***	

TABLE 2.1(b)

Method		P5	P6	P7
Gillett and Miller	–	1389/17	1266/7	937/10
Mole and Jameson	–	1545/17	1100/7	-
Christofides et al (tree search)	–	1386/17*	1092/7	816/10*
Christofides et al (2 Phase)	–	1418/17	1066/7	827/10
	R	1387/17^{+}	1051/7^{+}	829/10^{+}
Probol (Real)	R	1429/17	1273/7	949/10
Fisher and Jai Kumar	-	1420/17	-	824/10
Saving/P (Real)	-	1399/17	1049/7*	837/10
	R	1396/17	1068/7	834/10

a/b :	a: total length, b: number of routes.
* :	best integer solution.
+ :	best Real solution.
- :	calculation of the Euclidean distances is probably done by an integer rounded operation.
R :	calculation of the Euclidean distances is done by real operation.

CHAPTER 3

HEURISTICS DEVELOPED FOR MULTI-DEPOT VRP

3.1 INTRODUCTION

This chapter describes in detail, heuristics developed for solving multi-depot VRP. First, route improvement procedures are described. These procedures operate on a feasible solution of the multi-depot VRP and tries to improve the solution value by rearranging some customers. Combine procedure combines tworoutes, if combined route is feasible and results in saving. The 2-optimal procedure, which is based on r-optimal procedures (Lin and Kernighan[1973]), rearranges the customers in the same route if it results in saving. Shift customer procedure tries to achieve savings by shifting customers from one route to another. The split and join procedure splits two routes and makes two new routes by joining components taken from both the routes. The interchange segment procedure interchanges customer segments between two routes.

A total of five heuristics have been presented. First heuristic is based on Clarke and Wright's [1964] saving criteria. Second heuristic starts with N single customer routes. At each iteration those two routes are combined which results in maximum savings. Third heuristic starts a route by linking a depot with the nearest customer to it. Then at each iteration, partial completed routes are expanded by linking last customer of the route with the nearest unrouted customer, subject to feasibility conditions. Fourth and fifth heuristics also make starting routes in the same way as heuristic 3 does, except these do not check for feasibility conditions while expanding a partial route. In this process some routes are made which are infeasible. Fourth and fifth heuristics use different procedures to convert infeasible routes into feasible routes.

3.2 ROUTE IMPROVEMENT PROCEDURES

3.2.1 Combine Routes

All pairs of routes are checked for feasibility and savings, resulting from combining two routes into one. Routes, whose combining results in maximum saving and yields feasible combined route, are combined. This process is repeated until no more routes can be combined.

Step 1 Select route i for i = 1 to the number of routes.

Step 2 Select route j for j = 1 to the number of routes and $i \neq j$.

Step 3 Calculate savings if routes i and j are combined, as per the following orientations;

Route i $\longrightarrow$ a,...,b ; Route j $\longrightarrow$ x,...,y ;

dp i $\longrightarrow$ depot of route i; dp j $\longrightarrow$ depot of route j

Orientation	Saving
1. a,...,b,x,...,y, dp i	$c_{b,dp\ i} + c_{dp\ j,x} + c_{y,dp\ j} - c_{b,x} - c_{y,dp\ i}$
2. a,...,b,x,...,y, dp j	$c_{dp\ i,a} + c_{b,dp\ i} + c_{dp\ j,x} - c_{b,x} - c_{dp\ j,a}$

3.	a,...,b,y,...,x, dp i	$c_{b,dp\ i} + c_{y,dp\ j} + c_{dp\ j,x} - c_{b,y} - c_{x,dp\ i}$
4.	a,...,b,y,...,x, dp j	$c_{dp\ i,a} + c_{b,dp\ i} + c_{y,dp\ j} - c_{b,y} - c_{dp\ j,a}$
5.	b,...,a,x,...,y, dp i	$c_{dp\ i,a} + c_{dp\ j,x} + c_{y,dp\ j} - c_{a,x} - c_{y,dp\ i}$
6.	b,...,a,x,...,y, dp j	$c_{b,dp\ i} + c_{dp\ i,a} + c_{dp\ j,x} - c_{a,x} - c_{dp\ j,b}$
7.	b,...,a,y,...,x, dp i	$c_{dp\ i,a} + c_{y,dp\ j} + c_{dp\ j,x} - c_{a,y} - c_{x,dp\ i}$
8.	b,...,a,y,...,x, dp j	$c_{b,dp\ i} + c_{dp\ i,a} + c_{y,dp\ j} - c_{a,y} - c_{dp\ j,b}$

Mark the orientation and the depot which gives maximum savings.

Step 4 Check the feasibility of combining route i & j and if it is feasible record i and j with maximum saving, combining orientation and depot.

Step 5 j = j + 1 ; go to Step 2

Step 6 i = i + 1 ; go to Step 1

Step 7 Sort savings in order to find out the maximum saving routes i and j.

Combine i and j into one route as per the orientation for maximum saving, update number of routes.

Repeat the combine procedure until two routes can be combined any more.

3.2.2 2-Optimal Procedure

This is based on r-optimal routes method of Lin and Kernighan [1973]. A 2-exchange involves the substitution of two edges, say (i,i+1) and (j,j+1), with two other edges (i,j) and (i+1,j+1). The orientation of the path (i+1,...,j) is reversed in the new route.

Route $\longrightarrow$ d,1,2,3,...,i-1, i, i+1,...,j-1, j, j+1,...,k,d,

after a 2-exchange becomes

Route $\longrightarrow$ d,1,2,3,...,i-1, i, j, j-1,...,i+1, j+1,...,k,d.

Such an exchange results in a local improvement if and only if

$$c_{i,j} + c_{i+1,j+1} < c_{i,i+1} + c_{j,j+1}.$$

Step 0 Represent the route as an array whose first and last elements are depot and element 2 onward are the customers on the route in the same sequence.

$$\begin{array}{ccc} X[1], & X[2],\ldots\ldots, X[K-1], & X[K] \\ \downarrow & & \downarrow \\ \text{depot} & \longleftarrow\text{customers}\longrightarrow & \text{depot} \end{array}$$

Step 1 Set i = 1.

Step 2 Set j = i + 1.

Step 3 Test for 2-exchange for edges (i,i+1) and (j,j+1). If 2-exchange results in savings and if time window constraints are also specified, then test for feasibility of path reversal from (i+1,...j-1,j) to (j,j-1,...,i+2,i+1).

If feasible, then perform the exchange.

Step 4 j = j+1 ; go to Step 3 if j ≤ K-1.

Step 5 i = i+1 ; go to Step 2 if i ≤ K-2.

3.2.3 Shift Customer

This procedure reallocates the customers to other routes. Each customer is tested for inserting into all other routes. It is reallocated at a place where maximum saving is achieved, if any. This process is repeated for all the customers.

Step 1 Select route r for r = 1 to the number of routes.

Step 2 Select customer X[i] of route r for i = 1 to the number of customers in route r.

Step 3 Select route s for s = 1 to the number of routes and $s \neq r$.

Calculate saving for customer i in route r if it is placed in route s after customer y[j].

Decrease of length in route r

$$D = C_{X[i-1],X[i]} + C_{X[i],X[i+1]} - C_{X[i-1],X[i+1]}.$$

Increase of length in route s

$$I = C_{Y[j],X[i]} + C_{X[i],Y[j+1]} - C_{Y[j],Y[j+1]}.$$

saving = D - I.

If saving > 0 then check feasibility of inserting customer X[i] in route s, and if feasible, then record it.

Repeat Step 3 for all values of j corresponding to all values of s.

Step 4 - If there is a route and position where insertion of customer X[i] results in saving, delete customer X[i] from route r and insert it at the maximum saving position.

Step 5 - Repeat from Step 2 for all values of i

Step 6 Repeat from Step 1 for all values of s

3.2.4 Shift Segment

Some times, two or more customers form a cluster such that they are close to each other but far away from other customers. Shifting of one customer from this cluster to another route will not result in savings unless whole cluster is moved from one route to another. Shift segment procedure removes m-consecutive customers from a route and inserts them in a route, at such position which results in maximum saving, if any.

Let m consecutive customers are to be moved from one route to another. It is similar to shift procedure with the following modifications

Step 2 Select customers from X[i] to X[i+m-1] for i = 1 to the number of customer in route r -m + 1.

Step 3 Decrease of the length in route r

$$D = \sum_{K=i-1}^{K=i+m-1} C_{X[K],X[K+1]} - C_{X[i-1]X[i+m]}.$$

Increase of length in route r

$$I = \sum_{K=i}^{K=i+m-2} C_{X[K],X[K+1]} + C_{Y[j],X[i]} + C_{X[i+m-1],Y[j+1]}.$$
$$- C_{Y[j]\ Y[j+1]}.$$

3.2.5 Split and Interchange Components

This procedure picks two routes at a time, splits both into two components. New routes are made by joining two components (one from each route) in various combinations and are tested for savings. If these changes result in saving, new routes are made as per the combination which gives the maximum saving. Let there are two routes i and j;

Route i $\quad X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{k_i}$

Route j $\quad Y_1, Y_2, \ldots, Y_q, Y_{q+1}, \ldots, Y_{k_j}$

where k_i and k_j are the number of customers in route i and route j respectively. These routes are broken into two components (each) by removing arc from p to p+1 and q to q+1. New routes can be made in one of the following ways;

1 Route i' $\longrightarrow X_1, X_2, \ldots, X_p, Y_{q+1}, \ldots, Y_{k_j}$.

Route j' $\longrightarrow Y_1, Y_2, \ldots, Y_q, X_{p+1}, \ldots, X_{k_i}$.

2 Route i' $\longrightarrow X_1, X_2, \ldots, X_p, Y_q, Y_{q-1}, \ldots, Y_1$.

Route j' $\longrightarrow X_{k_i}, Y_{p+2}, Y_{p+1}, Y_{q+1}, Y_{q+2} \ldots Y_{k_j}$.

If depots for i and j are different, then the appropriate depots are to be find out for the new routes.

Step 1 Select route i, i = 1 to the number of routes.

Step 2 Select route j, i<>i and j = 1 to the number of routes.

Step 3 Select arc p to p+1 in route i. p=1 to the number of customers in route i - 1.

Step 4   Sleect arc q to q+1 in route j. q=1 to the number of customer in route i - 1.

Step 5   Calculate Savings

(i) Saving = $C\left[X_p, X_{p+1}\right] + C\left[Y_q, Y_{q+1}\right] - C\left[X_p, Y_{q+1}\right] - C\left[Y_q, Y_{p+1}\right]$

+ maximum of change in length because of depot reassignment.

(ii) Saving = $C\left[X_p, X_{p+1}\right] + C\left[Y_q, Y_{q+1}\right] - C\left[X_p, Y_q\right] - C\left[X_{p+1}, Y_{q+1}\right]$.

+ maximum of change in length because of depot reassignment.

If any of these is positive, record it with values of p, q, depot and joining combination.

Step 6   Go to Step 4 for the next value of q.

Step 7   Go to Step 3 for the next value of p.

Step 8   If there is a saving, make the new routes as per the combination, p and q values which results in the maximum saving.

Step 9   Go to Step 2 for the next value of j.

Step 10   Go to Step 1 for the next value of i.

3.2.6 Exchange Segments

This procedure picks two routes at a time, removes one segment of customers from each of the two routes and places these segments into routes after exchanging these with each other. Segment from route 1 goes to route 2 and from route 2 to route 1 in such a way, which results in maximum saving, if any. Let two routes are i and j.

Route i $X_1, \quad X_{p-1}, X_p, \ldots, X_q, X_{q+1}, \ldots, X_{k_i}$.

Route j $Y_1, \quad Y_{s-1}, Y_s, \ldots, Y_t, Y_{t+1}, \ldots, Y_{k_j}$.

Customer segments from X_p to X_q and Y_s to Y_t can be removed and exchanged as per the following ways.

Route i′ (a) $X_1, \ldots, X_{p-1}, Y_s, \ldots, Y_t, X_{q+1}, \ldots, X_{k_i}$.

(b) $X_1, \ldots, X_{p-1}, Y_t, \ldots, Y_s, X_{q+1}, \ldots, X_{k_i}$.

Route j′ (a) $Y_1, \ldots, Y_{s-1}, X_p, \ldots, X_q, Y_{t+1}, \ldots, Y_{k_j}$.

(b) $Y_1, \ldots, Y_{s-1}, X_q, \ldots, X_p, Y_{t+1}, \ldots, Y_{k_j}$.

Step 1 Select route i, i=1 to the number of routes.

Step 2 Select route j, j<>i and j=1 to the number of routes.

Step 3 Select segment in route i from p to q.

(a) p = 1 to the number of the customers in route i.

(b) q = p to the number of customers the in route i.

Step 4 Select segment in route j from s to t.

(a) s = 1 to the number of customers in route j.

(b) t = 1 to the number of customers in route j.

Step 5 Calculate savings in the following manner:

Route i (i) $\text{saving\_i} = C\left[X_{p-1}, X_p\right] + C\left[X_q, X_{q+1}\right]$

$$- \left[X_{p-1}, Y_s\right] - C\left[Y_t, X_{q+1}\right]$$

(ii) $\text{saving\_i} = c\left[X_{p-1}, X_p\right] + c\left[X_q, X_{q+1}\right]$

$- \left[X_{p-1}, Y_t\right] - c\left[Y_s, X_{q+1}\right]$

Route j (i) $\text{saving\_j} = c\left[Y_{s-1}, Y_s\right] + c\left[Y_t, Y_{t+1}\right]$

$- \left[Y_{s-1}, X_p\right] - c\left[X_q, Y_{t+1}\right]$

(ii) $\text{saving\_j} = c\left[Y_{s-1}, Y_s\right] + c\left[Y_t, Y_{t+1}\right]$

$- \left[Y_{s-1}, X_q\right] - c\left[X_p, Y_{t+1}\right]$

Saving = max {saving_i} + max {saving_j}

If saving is positive record it with the values of p,q,s,t and orientations for pq and st.

Step 6 (a) Go to Step 4(b) for the next value of t.

(b) Go to Step 4(a) for the next value of s.

Step 7 (a) Go to Step 3(b) for the next value of q.

(b) Go to Step 3(a) for the next value of p.

Step 8 If saving is positive, then exchange segments as per the orientation and values of p,q,s,t which results in maximum savings.

Step 9 Go to Step 2 for the next value of j.

Step 10 Go to Step 1 for the next value of i.

3.2.7 Sequence of calling route improvement procedures

(a) Shift customers.

(b) Combine routes.

(c) 2-optimize routes.

(d) Shift customers.

(e) Split and interchange components.

(f) 2-optimize routes.

(g) Shift customers.

(h) Interchange segments (Normally not used because of high computation time).

3.3 HEURISTIC 1

3.3.1 Construction of starting Routes

Step 1 Determine the nearest depot for all the N customers. Make N single customer routes, one for each customer. Each route consists of one customer and its nearest depot.

Route i : $\left\{d_k, X_i, d_k\right\}$ d_k : nearest depot to X_i

k : depot number

Step 2 Sort all the routes lengthwise in the descending order.

3.2.2 Merging single customer routes into other routes

This procedure picks customers from single customer routes, starting from largest length route. It then, inserts them at places which results in the maximum saving and deletes the corresponding single customer route.

Step 1 Pick the single customer route having highest length

Route i : Route i contains only a single customer. Route i is selected starting from the highest length route.

Step 2 Select route j ; for j = 1 to the number of routes and $i \neq j$.

Step 3 Calculate savings for inserting customer X_i of route i into route j after customer K

$$\text{Savings} = C_{d_i, x_i} + C_{x_i, d_i} + C_{X_{j[k]}, X_{j[k+1]}} - C_{X_{j[k]}, X_i} - C_{X_i, X_{j[k+1]}}$$

Step 4 If savings > 0, then check for feasibility of inserting X_i into j after K^{th} customer. If feasible, then record it.

Step 5 If there are route j into which X_i can be inserted without violating feasibility conditions and savings occur, then insert X_i into route j which results in maximum saving at the maximum saving position.

Step 6 Go to Step 1 and select next single customer route and repeat it until there are no more single customer routes which can be merged into other routes.

3.3.3 Sequence of Procedures

Route building and improvement procedures are called in the following sequence in Heuristic 1 :

(1) Construction of starting routes.

(2) Merging single customer routes into other routes.

(3) Route improvement procedures in the sequence as given in section 3.2.7.

3.4 HEURISTIC 2

This heuristic is almost similar to Heuristic 1 except that procedure 'merging single customer routes into other routes' is not called. We start with the construction of starting routes which makes N single customer routes. Then combine procedure is called. It combines two routes in one iteration after scanning all pairs of two routes out of total routes. In Heuristic 1, number of routes given to combine procedure are much less then N but in this heuristic exactly N routes are given to combine procedure which has to calculate savings for $\binom{N}{2}$ pairs and then two routes are combined. Hence it is not polynomial time bounded for N but results produced by it are good. Therefore, it is mainly used for comparison purposes.

3.4.1 Sequence of Procedures

Route building and improvement procedures are called in the following sequence for Heuristic 2:

(1) Construction of starting routes.

(2) Combine routes.

(3) Route improvement procedures in the sequence as given in section 3.2.7.

3.5 HEURISTIC 3

3.5.1 Building of Routes based on nearest neighbour approach

A vehicle may start from one of the depots, thus all depots form the set(S) of potential starting points for a route. For every member of set(S), i.e., for each depot, nearest customer is searched from set (U) of unrouted customers, and marked. In the beginning, set U contains all the customers. Let there are three depots d_1, d_2, d_3; X_1, X_2, X_3 are the customers nearest to d_1, d_2, d_3 respectively. Out of these three nearest customers (X_1, X_2, X_3) that one is selected for linking to its depot whose distance is minimum from its corresponding depot.

Select from (X_1, X_2, X_3) such that it is $\min\left\{C_{d_1,X_1}, C_{d_2,X_2}, C_{d_3,X_3}\right\}$. Let C_{d_1,X_1} is minimum. Hence X_1 is selected for starting a new route and it is deleted from set U.

Route 1 $\longrightarrow d_1, X_1, \ldots$ partial route

Now one vehicle is standing at X_1, from where it can go to any other customer. Hence X_1 is also a potential starting point for a vehicle; hence X_1 is also added to set S. Nearest customers to every member of set S (all depots and X_1) are searched and marked. Let they are as follows

$$d_1\text{-}X_4,\ d_2\text{-}X_2,\ d_3\text{-}X_3,\ X_1\text{-}X_5$$

Out of these nearest customers (X_4, X_2, X_3, X_5), we select the one whose distance is minimum among corresponding member of S.

$$\min\left\{C_{d_1,X_4},\ C_{d_2,X_2},\ C_{d_3,X_2},\ C_{X_1,X_5}\right\}$$

Let X_3 achieves this minimum. Now X_3 is selected for starting a new route and is deleted from U.

Route 2 $\longrightarrow d_3, X_3, \ldots$ partial route

Now, X_3 also becomes a potential point from where a vehicle can go to visit another customer, hence X_3 is added to set S. Again, nearest customers to all the members of set S (all depot and X_1, X_3) are searched and marked. Let they are as follows:

$$d_1-X_4,\ d_2-X_2,\ d_3-X_6,\ X_1-X_5,\ X_3-X_2$$

A customer may be nearest to more then one member of set S, as X_2 is nearest to d_2 and X_3. Out of these nearest customers $(X_4, X_2, X_6, X_5, X_2)$ that one is selected whose distance is minimum

$$\min \left\{ C_{d_1,X_4},\ C_{d_2,X_2},\ C_{d_3,X_6},\ C_{X_1,X_5},\ C_{X_3,X_2} \right\}$$

Let C_{X_1,X_5} is minimum. Hence X_5 is selected for linking. This time it is nearest to a customer X_1. Route containing X_1 is expanded to include X_5.

Route 1 $\quad d_1, X_1, X_5, \ldots$

We delete X_5 from U. Now vehicle of route 1 has reached to X_5, therefore, X_1 is no more a potential starting point from where any other customer can be visited. X_1 is deleted from the set S and X_5 is added to the set S.

Before expanding a partial route, feasibility conditions are to be checked for adding a customer to the partial route.

Suppose it is not feasible to add X_5 to route 1. Then out of the nearest customers, $(X_4, X_2, X_6, X_5, X_2)$ that customer is selected whose distance is second minimum. Since there are some customers who are nearest to the depot, ultimately new route can always be started which has to be feasible.

After adding X_5 to Route 1, set S contains $(d_1, d_2, d_3, X_5, X_3)$ and the process is repeated until all the customer in U are included in some route. It is summarized as follows.

At any point of time, set S contains all depots and last customers of all the partial routes. Nearest neighbours to all the members are searched and the one whose distance is minimum is linked with the corresponding member. If corresponding member is a depot, new route is started. If it is a last customer of some route, the route is expanded to include this customer, subject to feasibility conditions. This process is repeated until all the customers are routed. In the end, all partial routes are completed by linking last customer of the partial routes to its depot.

Step 1 Define set S, set of potential starting points, as the set of all depots.

$S = \left\{ \text{all depots } d_i, \quad i = 1 \text{ to } M \right\}.$

In the actual implementation, depots are represented by X_i, $i=N+1$ to $N+M$ and

s_K is the Kth member of set S.

Define set U, set of unrouted customers, equal to all customers:

$U = \left\{ \text{all customers } x_i \quad i=1 \text{ to } N \right\}$

Step 2 Find out nearest customer to every member of S.

Nearest customer to $s_K = y_K = \left\{ x_i : C_{s_K, X_i} \right.$ is minimum for all $\left. x_i \in U \right\}$

Step 3 Select that pair of s_k , y_k out of those from Step 2 which has minimum distance C_{s_k,y_k} for all (s_k,y_k) pairs identified in Step 2.

$$\min_{s_k \in S} \left\{C_{s_k,y_k}\right\}$$

Step 4a If for minimum distance pair (s_k,y_k), s_k happens to be a depot, then do the following: start a new route r with s_k as depot and y_k as customer

route r $\longrightarrow s_k, y_k, \ldots$

$$U = U - \left\{y_k\right\}$$

$$S = S + \left\{y_k\right\}.$$

Go to Step 2 until U has a customer.

Step 4b If for minimum distance pair (s_k,y_k), s_k happens to be a customer which is the last customer of say route r, then test for feasibility of adding y_k to route r

If feasible then do following:

Expand route r by adding y_k to it.

Route r $\longrightarrow d_r, X_i \ldots, s_k, y_k$

$$U = U - \left\{y_k\right\}$$

$$S = S - \left\{s_k\right\} + \left\{y_k\right\}.$$

Go to Step 2 until U has a customer.

If addition of y_k to route r is not feasible then find out next minimum pair (s_1, y_1), i.e., minimum distance pair after considering (s_k, y_k) is not in the list. Repeat Step 4.

Step 5 When U is empty, complete all partial routes by linking last customer to its depot.

3.5.2 Sequence of Procedures

Route building and improvement procedures are called in the following sequence in Heuristic 3.

(1) Build routes based on nearest neighbour approach.

(2) Route improvement procedures in the sequence as given in Section 3.2.7.

3.6 HEURISTIC 4

Heuristic 3 work very well if the route constraints (capacity, length or time) are not there or are loose. Its solution value comes out to be the lowest in comparison to other heuristics. But as constraints become tight, performance of Heuristic 3 degrades. It looks that routes building procedure is very good for unconstraint case but not good for tight constraints.

In Heuristic 4, routes are built without considering feasibility conditions, i.e., routes are expanded irrespective of whether addition of customer makes it feasible or infeasible. Another procedure is used which makes infeasible routes, feasible by breaking it into several feasible routes.

3.6.1 Building of routes without considering feasibility

This procedure is similar to procedure 'building of routes based on nearest neighbour approach' of Section 3.5.1, except while expanding a partial route feasibility conditions are not tested.

Step 4b At this step feasibility testing is not done.

3.6.2 Converting infeasible route into feasible routes

Each infeasible route is broken into several smaller feasible routes. Let r is an infeasible route. Start from the first customer of r, travel along the. route till the first customer which introduces infeasibility is found, say (i+1)th customer. Now remove the segment starting from first customer to the ith customer from route r and make another route r_1 having the depot of route r and customers from 1 to i. Route r_1 will be feasible because infeasibility is caused by (i+1)th customer which we have not taken.

Route r —— $d,\underbrace{1,2,\ldots i-1,i}_{},i+1\ldots$

After removing 1 to i customer

Route r d, i+1,...

Route r_1 d, 1,2,...i-1, i, d

Segment d,1,...i was feasible in route r but in route r_1 distance $c_{X[i],d}$ is added which may make it infeasible. Then i or as many customers are returned to route r to make r_1 feasible. Now,if route r is feasible then stop. If route r is still infeasible, again repeat the same process,i.e., remove segment of as many customers so as to make another feasible route and repeat this process until r becomes feasible.

Let infeasible route r is having depot d and customers X[1], X[2], X[3] ...

Step 1 Find out maximum value of i such that route made from first i customers of route r is feasible

$$C_{d,X[1]} + \sum_{K=1}^{i-1} C_{X[K],X[K+1]} + C_{X[i],d} \leq \text{maximum allowed length of route}$$

$$\sum_{K=1}^{i} q_K \leq \text{capacity of vehicle}$$

Largest value of i is taken such that all the feasibility conditions are satisfied.

Step 2 Remove first i customers from the route r and make another route r_1 with these i customers. Assign that depot to this route which results in minimum length of the route:

route r d, X[i+1], X[i+2]...

route r_1 d, X[1], X[2],...,X[i],d

Step 3 Test route r for feasibility.

If infeasible, repeat Step 1 until route r becomes feasible.

Step 4 Reassign depots to each route such that route length is minimum.

3.6.3 Sequence of Procedure

Route building, feasible making and route improvement procedures are called in the following sequence.

(1) Build routes based on nearest neighbour approach without considering feasibility.

(2) 2-optimize all the routes.

(3) Break infeasible routes into feasible routes.

(4) Route improvement procedure in the sequence as given in Section 3.2.7.

3.7 HEURISTIC 5

This heuristic is similar to Heuristic 4. In this heuristic also, first of all routes are made using nearest neighbour approach, without considering for feasibility. A different approach is used to break infeasible routes into feasible routes.

3.7.1 Converting infeasible route into feasible routes by removing largest arc

Each infeasible route is broken into two routes by removing the largest arc from the route. Open ends are connected to the depot. In this process, one or both components may become feasible. Infeasible component are broken into two routes by removing largest arc. This process is repeated until all routes are feasible

Let L is the list of infeasible routes.

Step 1 Pick any infeasible route r

Route r $\longrightarrow$ d, X[1], X[2],...X[i], X[K],d

Find out the largest arc in the route from X[1] to X[K]

$$\max_{i=1 \text{ to } K} \left\{ C_{X[i],\ X[i+1]} \right\}$$

Step 2 Remove arc X[i] to X[i+1] and make two routes

r_1 —— d, X[1], X[2]... X[i],d,

r_2 —— d, X[i], X[i+1]..X[K],d.

Remove route r from the list L.

Step 3 Test for feasibility of r_1 and r_2

If r_1 is infeasible, add r_1 to the List L.
If r_2 is infeasible, add r_2 to the List L.

If both are infeasible then both are added to the List L.

Step 4 If List L contains an infeasible route, pick one of these and go to Step 1. Repeat this process until list L is empty.

Step 5 Assign each route that depot which results in minimum length of the route.

3.7.2 Sequence of Procedure

Route building, feasible making and route improvement procedures are called in the following sequence.

(1) Build routes based on nearest neighbour approach without considering feasibility.

(2) 2-optimise all the routes.

(3) Break infeasible routes into feasible routes.

(4) Route improvement procedures in the sequence as given in Section 3.2.7.

CHAPTER 4
COMPUTATIONAL RESULTS

4.1 INTRODUCTION

In this chapter, we present the computational results for the heuristics developed by us. Solution values and computational times are given for test problems taken from literature and also for some randomly generated test problems. Solution values, after initial construction of routes and after applying each route improvement procedure, are also given for Heuristic 1 and Heuristic 5. In the end, a comparison is made between best published results of test problems 1-7 and results of our best heuristics.

4.2 COMPUTATIONAL RESULTS

All the heuristics and route improvement procedures have been coded in PASCAL and tested on HP9000/850 computer. Data for test problems 1-10 has been taken from literature whose details are given at Section 2.7. Problem 11-20 are randomly generated problems. The X and Y coordinates for customers and depots are generated by random number generator in the range of 0 to 100, customer demands in the range of 0 to 50. Other details of these problems are also given in Section 2.7.

The solution values are given in Table 4.1 for problems 1-10 and in Table 4.3 for problems 11-20. Solution values after initial construction of routes and after applying all the route improvement procedures, are both given. The computation times in seconds on HP9000/850 computer are given in Table 4.2 for problems 1-10 and in Table 4.4 for problems 11-20.

Solution values after each route improvement procedures, are presented in Table 4.5 for Heuristic 1 and in Table 4.6 for Heuristic 5. Solution improved by Interchange Segments procedure is also presented. We have not included this procedure in all other computations because of its excessive computation time. In the end, results of Heuristic 1 and 5 have been compared with best published results in Table 4.7.

4.3 CONCLUSIONS

We have presented five heuristics for solving the multi-depot vehicle routing problem. The results of Heuristic 1, Heuristic 2 and Heuristic 5 are comparable with the best published results for problems 1-7. The computational times for Heuristic 2 are not polynomial bounded which is clear from Table 4.2. Therefore, we do not consider it as a good heuristic. Heuristic 5 shows better performance on Test problems taken from literature and some of the solution values are even lower than the best published solution.

The testing of heuristics on randomly generated large size problems, clearly shows supremacy of Heuristic 1 over Heuristic 5. However, as clear from the computational time graph, Heuristic 1 takes more time than Heuristic 5 and also its rate of growth is higher. The computational times are lowest for Heuristic 3 but results are marginally higher than Heuristic 1.

Out of route improvement procedures, Shift customer procedure does reasonably good improvements. Other procedures are not very effective. Interchange segment procedure does good improvement for some of the problems but it increases computational time substantially.

All these conclusions are based on testing of the heuristics on test problems and randomly generated problems. We have not done any theoretical analysis for the behaviour of the heuristics.

Table 4.1

Accuracy Analysis of VRP Heuristics

	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Heuristic 5
P1 + 50,1	580.9 * 566.9	585.7 566.0	639.6 548.5	692.1 559.2	920.4 551.7
P2 75,1	919.3 893.0	910.7 898.6	1069.7 912.9	1058.3 914.9	1157.9 875.1
P3 100,1	946.4 882.7	895.5 883.3	1078.2 858.6	1084.8 894.2	1326.1 858.5
P4 150,1	1269.8 1072.2	1129.1 1103.0	1343.4 1150.0	1318.6 1134.0	1871.3 1082.2
P5 199,1	1661.3 1388.2	1391.0 1377.9	1697.3 1448.4	1770.5 1453.1	2239.8 1410.0
P6 120,1	1611.3 1213.4	1076.9 1047.0	1373.7 1199.7	1371.2 1184.4	1163.9 1055.1
P7 100,1	1025.7 896.6	834.3 822.4	1115.6 911.8	1045.8 866.7	974.8 826.2
P8 199,5	1301.8 1111.1	1145.8 1111.4	1466.5 1137.7	1457.2 1144.3	1580.5 1186.4
P9 120,5	1149.3 811.1	786.1 737.8	893.1 725.4	843.4 732.4	879.7 796.8
P10 100,2	933.0 798.3	800.9 768.4	976.5 809.9	965.8 814.9	8767.7 767.8

+ : Problem number
number of customers, number of depots.

* : Route length after construction heuristic,
Route length after applying route improvement procedures.

Table 4.2

Time Analysis

	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Heuristic 5
P1 50,1	1.7 +	4.4	0.6	0.8	0.7
P2 75,1	2.8	12.9	1.2	1.0	1.3
P3 100,1	7.5	32.0	2.6	3.6	3.5
P4 150,1	19.9	104.5	4.3	6.6	6.5
P5 199,1	43.4	239.8	7.6	9.4	11.4
P6 120,1	10.8	53.8	1.7	4.1	2.7
P7 100,1	7.8	33.2	1.8	2.9	3.2
P8 199,5	21.3	249.0	9.8	11.1	16.0
P9 120,5	8.2	56.9	4.9	4.7	5.6
P10 100,2	6.5	34.2	2.9	3.0	3.3

+ Computation time in Seconds on HP9000/850 computer.

Table 4.3

Accuracy Analysis on Random Problems

	Heuristic 1	Heuristic 3	Heuristic 4	Heuristic 5
P11 + 100,5	1339.0 * 1185.2	1395.6 1200.9	1448.7 1250.6	1567.4 1186.7
P12 200,6	2255.8 2059.8	2465.4 2176.4	2601.6 2107.6	3025.3 2103.7
P13 300,4	3711.2 3408.9	3933.8 3493.8	3886.3 3483.9	5496.7 3465.8
P14 400,10	3226.0 2889.3	3507.7 2998.3	3525.3 3050.7	4544.1 2889.9
P15 500,4	5374.0 5099.7	5938.8 5300.0	5694.1 5188.5	8655.5 5424.3
P16 600,7	4166.2 3616.4	4232.2 3664.4	4143.2 3650.0	5571.8 3674.5
P17 700,5	5479.9 5163.1	5981.1 5445.8	5777.1 5175.3	8672.5 5239.0
P18 800,10	4380.8 3818.0	4589.0 3952.0	4578.7 3911.7	5784.1 3869.5
P19 900,10	4981.4 4056.6	4907.3 4249.3	4880.3 4247.9	6003.1 4073.6
P20 1000,20	4880.8 4046.2	4928.5 4184.8	5003.2 4245.4	6307.6 4157.0

+ : Problem number,
number of customers, ;number of depots.

* : Route length after construction heuristic,
Route length after applying route improvement procedures.

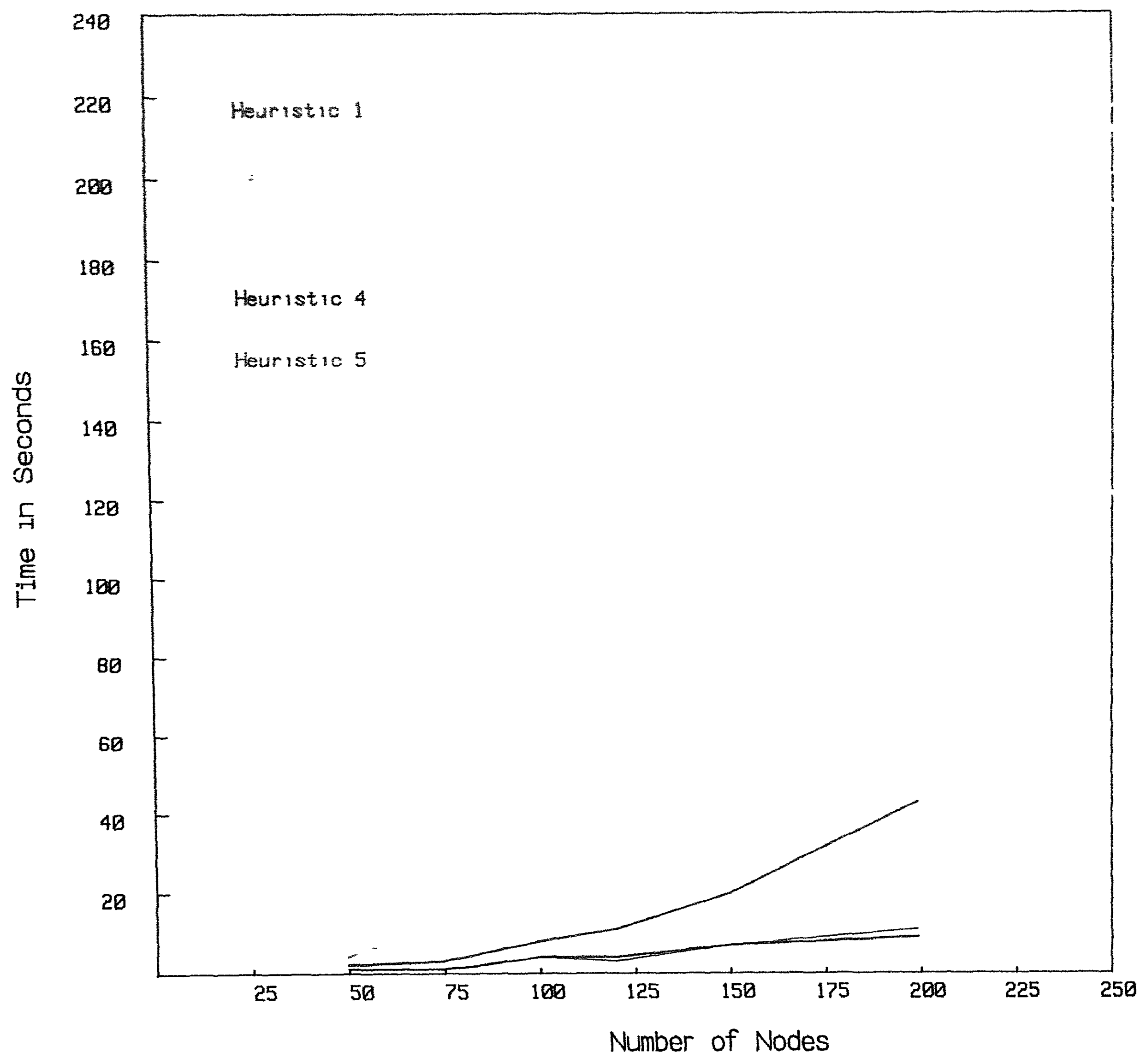
Table 4.4

Time Analysis

	Heuristic 1	Heuristic 3	Heuristic 4	Heuristic 5
P11 100,5	4.7 +	3.1	3.0	3.4
P12 200,6	17.4	10.6	9.3	12.9
P13 300,4	61.8	17.9	18.5	23.3
P14 500,10	73.9	37.2	35.4	49.5
P15 500,4	128.1	54.1	53.0	71.0
P16 600,7	221.8	70.3	83.1	110.9
P17 700,5	465.7	81.5	104.1	140.8
P18 800,10	375.0	163.7	185.2	220.2
P19 900,10	440.3	204.4	197.0	246.0
P20 1000,20	606.2	272.4	314.8	403.3

+ : Computation time in seconds on HP9000/850 computer.

Computational Time Analysis



Computational Time Analysis

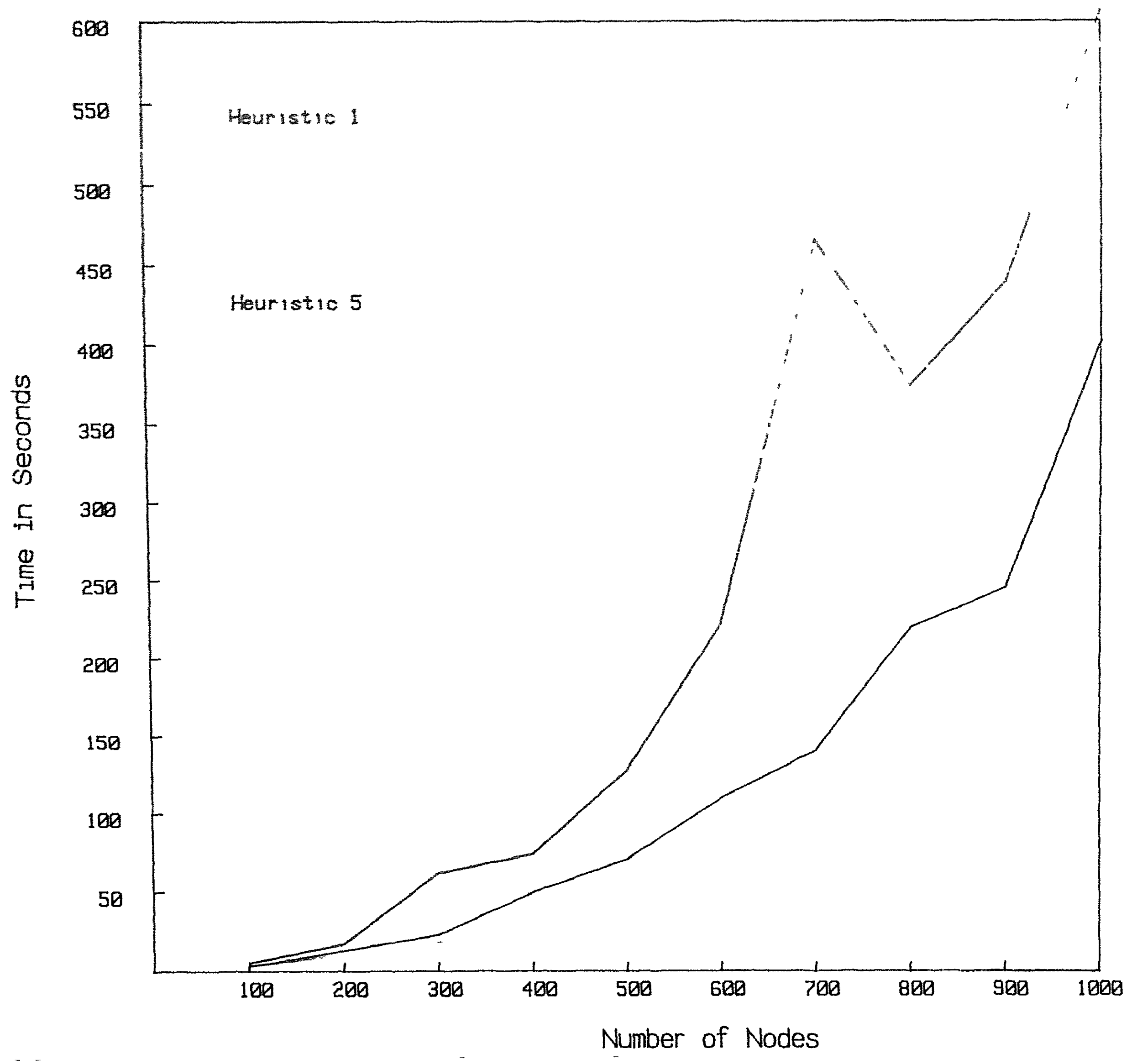


Table 4.5

Route Improvements for Heuristic 1

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7	Stage 8
P1	580.9	567.7	567.7	566.9	566.9	566.9	566.9	566.9
P2	919.3	893.1	893.1	893.1	893.1	893.0	893.0	893.0
P3	946.4	898.6	898.6	884.5	882.7	882.7	882.7	882.7
P4	1269.8	1131.8	1087.5	1080.0	1078.9	1076.2	1072.2	1072.2
P5	1661.3	1487.1	1434.6	1418.0	1415.2	1398.6	1395.7	1388.2
P6	1611.3	1324.5	1305.3	1279.5	1276.8	1241.8	1223.8	1213.4
P7	1025.7	955.0	955.0	950.5	946.8	915.0	899.4	896.6
P8	1301.8	1191.0	1153.2	1144.2	1140.3	1136.3	1129.1	1111.1
P9	1149.3	1047.4	861.2	817.1	815.1	814.0	811.4	811.1
P10	933.0	912.0	857.0	847.9	830.2	830.0	828.4	798.3

Stage 1 :	Starting routes;
Stage 2 :	Shift customers;
Stage 3 :	Combine routes;
Stage 4 :	2-optimal;
Stage 5 :	Shift customers;

Table 4.6

Route Improvements for Heuristic 5

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7	Stage 8
P1	920.4	551.7	551.7	551.7	551.7	551.7	551.7	551.7
P2	1157.9	940.8	940.8	927.4	881.1	880.5	876.6	875.1
P3	1326.1	944.5	935.1	917.7	894.2	881.9	881.9	858.5
P4	1871.3	1199.3	1199.3	1168.3	1090.8	1090.3	1089.6	1082.2
P5	2239.8	1508.8	1457.7	1444.3	1414.3	1413.2	1412.8	1410.0
P6	1163.9	1097.2	1087.4	1055.1	1055.1	1055.1	1055.1	1055.1
P7	947.9	856.5	829.8	826.2	826.2	826.2	826.2	826.2
P8	1580.5	1206.8	1206.8	1191.0	1187.2	1186.4	1186.4	1186.4
P9	879.7	826.3	819.3	803.7	803.7	799.7	796.8	796.8
P10	876.0	795.9	789.5	767.8	767.8	767.8	767.8	767.8

Stage 6 : Split and join routes;
Stage 7 : 2-optimize;
Stage 8 : Shift customers;
Stage 9 : Interchange segments.

Table 4.7

Comparison with best published result

	P1	P2	P3	P4	P5	P6
Best Published Real Result	525	844	832	1082	1387	1051
Christotide et al (2 phase)	537	873	858	1082	1387	1051
Heuristic 1	566	893 880*	882 877*	1072 1068	1388	1213 1186*
Heuristic 5	551	875	858 840*	1082 1063*	1410 1384*	1055

* Route length after applying Interchange segments procedure.

REFERENCES

AHN, BYONG HUN AND SHIN, JAE-YEONG 1991. Vehicle routing with time windows and time varying congestion, *Journal of Operational Research Society*, 42, 5, 393-400.

ALI, A. I., AND KENNINGTON, J.L. 1990. The M-TSP: A duality based branch and bound algorithm, *Technical Report OR 80018, Southern Methodist University, Dallas*, TX.

BODIN, L.D. 1990. Twenty years of routing and scheduling *Operations Research*, 38, 4, 571-579.

BODIN, L. AND GOLDEN. B. 1981. Classification in vehicle routing and scheduling, *Networks*, 11, 97-108.

BODIN, L. AND BERMAN, L. 1979. Routing and scheduling of school buses by computer, Transportation Science, 13, 113-129.

BODIN, L. AND KURSH, S. 1978. A computer assisted system for the routing and scheduling of street sweepers, *Operations research*, 26, 525-537.

BELLMORE, M. AND HONG, S. 1974. Transformation of multisalesman problem to the standard TSP, *Journal of ACM*, 21, 500-504.

CHRISTOFIDES et. al. 1979. *Combinatorial Optimization, John Wiley & Sons, New York.*

CHRISTOFIDES, N. AND EILON, S. 1969. An algorithm for the vehicle despatching problem, *Operational Research Quarterly*, 20, 309-318.

CHRISTOFIDES, N., MINGOZZI, A. AND TOTH, P. 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations, *Mathematical Programming*, 20, 255-282.

CHRISTOFIDES, N. 1976. The Vehicle Routing Problem, *RAIRO Recherhe Operationelle,* 10, 55-70.

CHRISTOFIDES, N. 1976. Worst case analysis of a new heuristic for travelling salesman problem, *Report 388,* Graduate School of Industrial Administration, Carnegie Mellon University.

CROWDER, H. AND PADBERG, M. 1980. Solving large-scale symmetric TSP to optimality, *Operations Research*, 12, 568-581.

COOK, S.A. 1971. The Complexity of Theorem proving Procedures. *Proc. 3rd Annual ACM Symp on Theory of Computing*. Association for Computing Machinery, New York, 151-158.

CLARKE, G. AND WRIGHT, J. W. 1964. Scheduling of vehicles from a central depot to a number of delivery points, *Operation Research,* 12, 568.

DANTZIG, G. B. AND RAMSER, J. H. 1959. The truck dispatching problem, *Management Science*, 6, 80-90.

FISHER, M.L. AND JAIKUMAR, R. 1981. A generalized assignment heuristic for Vehicle routing, *Networks*,11, 109-124.

GARCIADIAZ, A. 1985. A heuristic circulation network approach to solve the multi-travelling salesman problem, *Networks*,15, 455-467.

GASKELL, T. J. 1967. Basis for vehice fleet scheduling *Operations Research*, 22, 340-349.

GILLET, B. E. AND MILLER, L. R. 1974. A heuristic algorithm for the vehicle dispatch problem, *Operation Research,* 22, 340.

GILLET, B. E. 1976. Vehicle dispatching : Sweep algorithm and extensions, ORSA-TIMS National Meeting Miami.

GILLET, B. AND JOHNSON, T. 1976. Multi-terminal vehicle dispatch algorithm, *Omega*, 4, 711-718.

HELD, M. AND KARP, R. 1971 The travelling-salesman problem and minimum spanning trees. Part II, *Mathematical Programming*, 1, 6-25.

HINSON, J. AND MULHERKAR, S. 1975. Improvements to the Clarke and Wright algorithm as applied to an air line scheduling problem, *Technical Report, Federal Express Corp, USA.*

KARP, R.M. 1972. Reducibility among combinatorial problems, Tech. Report # 3 University of California, Berkeley, California.

KARP, R. M. 1977. Probabilistic analysis of partitioning algorithms for the TSP in the plane, *Mathematics of Operations Research*, 2, 204-224.

KULKARNI, R. V. AND BHAVE, P. R. 1985. Integer Programming formulation of Vehicle Routing Problems, *European Journal of Operational Research,*20, 58-67.

LAPORTE, G. AND NOBERT 1979. A cutting plane algorithm for the M-salesman problem, *Raport De Reeherche No79-08, Montreal, Canada.*

LENSTRA, J. L. AND RINOOYKAN, A. H. G. 1981. Complexity of vehicle routing and scheduling Problems, *Networks*, 11, 221-227.

LIN, S. AND KERNIGHAN, B.W. 1973. An effective heuristic algorıthm for the TSP, *Operatıons Research*, 21, 498-516.

MAGNANTI, T. L. 1981. Combinatorial optimization and vehicle fleet plannıng : Perspectives and prospects, *Networks*, 11, 179-213.

MOLE, R. H. AND JAMESON, S. R. 1976. A sequential route building algorithm employıng a generalised savıng crıterıon, *Operatıonal Research Quarterly*, 27, 503.

NEWTON, R. AND THOMAS, W. 1974. Bus routing in a multı-school system, *Computer and Operations Research*, 1, 213-222.

NORBACK, J. AND LOVE, R. 1977. Heuristic geometric approaches to solving the TSP, *Management Scıene*, 23, 1208-1223.

ORIOFF, C. S. 1974. Routing a fleet of M vehicles to/from a central facility, *Networks*, 4, 147-162.

PAESSENS, H. 1988. The savings algorithm for vehicle routing problem, *European Journal of Operational Research*, 34, 336-344.

RUSSEL, R. 1977. An effective heuristic for the M-tour TSP with some side conditions, *Operations Research*, 25, 517-524.

SALHI, S. AND RAND, G.K. 1987. Improvements to vehicle routıng heuristics, *Journal of Operational Research Socıety*, 38, 3, 293-295.

SAVELSBERGH, M. W. P. 1990. An efficient implementation of local search algorithms for constrained routing problems, *European Journal of operational Research*, 47, 75-85.

STEWART, W. AND GOLDEN, B. 1979. A vehicle routing algorithm based on generalized Lagrange multipliers, *Proceedings of the AIDS Annual Convention*, New Orleans LA, 2, 108-110.

SUETSKA, J. A. AND HUCKFELDT, V. E. 1973. Computational experience with an M-saleman TSP algorithm, *Management Science*, 19, 790-799.

YELLOW, P. 1970. A computatioanl modifiation to the savings method of vehicle scheduling, *Operations Research,* 21, 281.